POLYNOMIAL PRECONDITIONED ARNOLDI WITH STABILITY CONTROL*

MARK EMBREE†, JENNIFER A. LOE‡, AND RONALD B. MORGAN§

Abstract. Polynomial preconditioning can improve the convergence of the Arnoldi method for computing eigenvalues. Such preconditioning significantly reduces the cost of orthogonalization; for difficult problems, it can also reduce the number of matrix-vector products. Parallel computations can particularly benefit from the reduction of communication-intensive operations. The GMRES algorithm provides a simple and effective way to generate the preconditioning polynomial. For some problems high degree polynomials are especially effective, but they can lead to stability problems that must be mitigated. A two-level "double polynomial preconditioning" strategy provides an effective way to generate high-degree preconditioners.

Key words. eigenvalues, polynomial preconditioning, Arnoldi, GMRES

AMS subject classifications. 65F15, 15A18

1. Introduction.

We seek eigenvalues and eigenvectors of a large (possibly non-symmetric) $n \times n$ matrix A. The restarted Arnoldi algorithm [28, 34] (invoked by MATLAB's `eigs` command) is a standard workhorse for such problems, but for some matrices convergence is slow. One can improve convergence via a shift-invert transformation, i.e., applying the algorithm to $(A - \mu I)^{-1}$ to find eigenvalues near $\mu \in \mathbb{C}$. Here we investigate an effective alternative that does not need any explicit inversion of A. Our new *polynomial preconditioned Arnoldi method* is fairly simple to implement and can accelerate convergence for difficult problems.

When applied to the matrix A and starting vector v, the Arnoldi algorithm approximates eigenvalues using Rayleigh–Ritz estimates from the Krylov subspace

$$\mathcal{K}_m(A, v) \equiv \operatorname{span}\{v, Av, \ldots, A^{m-1}v\}. \tag{1.1}$$

Any vector x in this space, including the approximate eigenvectors, can be written as $x = \omega(A)v$ for some $\omega \in \mathcal{P}_{m-1}$, where $\mathcal{P}_s$ denotes the set of polynomials of degree s or less. The Arnoldi process builds an orthonormal basis for the subspace (1.1) via a Gram–Schmidt process, requiring many inner products as m grows.

Polynomial preconditioning methods [15, 16, 27, 29, 30, 32, 37] apply the Arnoldi algorithm to the matrix $\pi(A)$, for some polynomial $\pi \in \mathcal{P}_d$. Now eigenvalue estimates are drawn from the Krylov subspace

$$\mathcal{K}_m(\pi(A), v) = \operatorname{span}\{v, \pi(A)v, \ldots, \pi(A)^{m-1}v\}, \tag{1.2}$$

*The first author was supported by NSF grant DMS-1720257. The third author was supported by NSF grant DMS-1418677.

†Department of Mathematics and Computational Modeling and Data Analytics Division, Academy of Integrated Science, Virginia Tech, Blacksburg, VA 24061 (`embree@vt.edu`).

‡Center for Computing Research, Sandia National Laboratories (`jloe@sandia.gov`).
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

§Department of Mathematics, Baylor University, Waco, TX 76798-7328 (`Ronald_Morgan@baylor.edu`).

a subspace of $\mathcal{K}_{d(m-1)+1}(A, v)$. The large subspace $\mathcal{K}_{d(m-1)+1}(A, v)$ contains better approximations to the desired eigenvectors of A than does $\mathcal{K}_m(A, v)$. A polynomial preconditioner π is effective if the low-dimensional subspace $\mathcal{K}_m(\pi(A), v) \subseteq \mathcal{K}_{d(m-1)+1}(A, v)$ contains such improved estimates of the desired eigenvectors.

More specifically, any $x \in \mathcal{K}_m(\pi(A), v)$ can be written as $x = \omega(\pi(A))v$, where $\omega \in \mathcal{P}_{m-1}$. Since $\omega \circ \pi \in \mathcal{P}_{d(m-1)}$, polynomial preconditioning leads to eigenvector estimates that are high-degree polynomials in A. This feature comes at a cost, since $\pi(A)$ must be applied to a vector each time the subspace dimension m is increased. In typical high-performance computing environments these matrix-vector products can be evaluated more efficiently than inner products, which require significant communication and synchronization. Thus the subspace (1.2) can be constructed much more efficiently (in terms of both work and storage) than building out a standard Krylov subspace (1.1) of dimension $d(m-1)+1$. In summary, polynomial preconditioning gives an efficient way to involve high-degree polynomials, while controlling the dimension of the subspace and limiting the cost of orthogonalization.

What is a good choice for the preconditioning polynomial π? Section 2 describes one choice for π, inspired by the GMRES algorithm. By the spectral mapping theorem, any eigenvalue λ of A is mapped to an eigenvalue $\pi(\lambda)$ of $\pi(A)$. As the convergence theory in section 3 indicates, effective preconditioners separate the desired eigenvalues from the undesired ones.

Polynomial preconditioning is a special kind of *spectral transformation*, in which the Arnoldi algorithm is applied to $f(A)$ for some function f that maps the desired eigenvalues of A to the largest magnitude eigenvalues of $f(A)$; see, e.g., [19]. Typically such transformations involve a matrix inverse. One might seek a polynomial preconditioner π that mimics a more complicated $f(A)$. For example, Thornquist [38] advocates $\pi(A) \approx (A - \mu I)^{-1}$. Here we do not seek π that approximates $(A - \mu I)^{-1}$, but merely one that distances the desired eigenvalues from the rest of the spectrum.

Although methods for polynomial preconditioning have been proposed in the past, they are not generally used in practice. *To become popular, a polynomial preconditioner must be both effective and easy to implement.* We shall also explore stability, an important consideration for practical algorithms.

Section 2 discusses the choice of the GMRES (minimum residual) polynomial for the preconditioning, followed by some convergence theory in Section 3. Numerical experiments begin in Section 4, and suggest several practical issues that a robust algorithm should address. Section 5 studies the sensitivity of π to the choice of GMRES starting vector, while Section 6 introduces a "damped" polynomial and discusses when it will be useful. Section 7 addresses numerical stability, showing how the addition of duplicate roots in π can control distant unwanted eigenvalues. Finally, Section 8 describes *double polynomial preconditioning*, which enables the use of very large degree polynomials. Throughout $\|\cdot\|$ denotes the vector and matrix 2-norm.

2. Minimum residual polynomials. We seek the eigenvalues of A nearest the origin.[1] For the polynomial preconditioner π we use the minimum residual polynomial [18, 23] that arises when solving the linear system $Ax = b$ using the GMRES

[1] If one seeks eigenvalues near $\mu \in \mathbb{C}$, replace A with $A - \mu I$. If μ is in the interior of the spectrum, note that $|\pi(z)|$ is less likely to attain its maximum over the spectrum at μ. The challenge of computing interior eigenvalues arises in Example 2, and is the subject of future work.

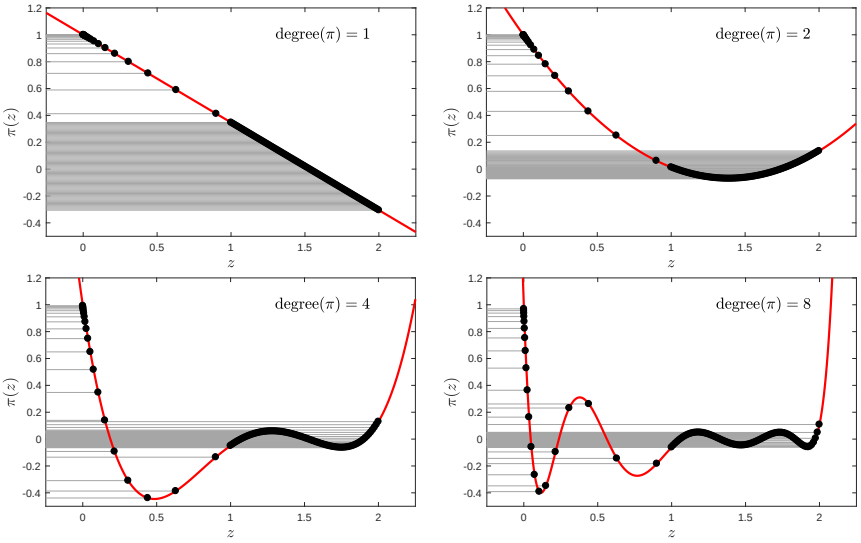

FIG. 2.1. *GMRES polynomials $\pi(z)$ (red lines) tend to separate eigenvalues closest to the origin, while mapping large-magnitude eigenvalues of A close to zero. The black dots and horizontal gray lines show the values of $\pi(\lambda_j)$.*

algorithm [33] (or MINRES for symmetric A [25]). This polynomial satisfies

$$\|\pi(A)b\| = \min_{\substack{p\in\mathcal{P}_d \\ p(0)=1}} \|p(A)b\|,$$

and hence $\pi \in \mathcal{P}_d$ must satisfy $\pi(0) = 1$; $|\pi(z)|$ will generally be small over the spectrum of A.[2] Denote the eigenvalues of A as $\sigma(A) = \{\lambda_j\}$, so the eigenvalues of $\pi(A)$ are $\{\pi(\lambda_j)\}$. If GMRES converges quickly, then $\|\pi(A)b\|$ is small, and the eigenvalues of $\pi(A)$ are typically concentrated near 0. However, the condition $\pi(0) = 1$ means that π is generally not able to map the small eigenvalues of A as close to zero, making these eigenvalues better separated in the spectrum of $\pi(A)$. Figure 2.1 illustrates this point, using a symmetric A with 20 eigenvalues logarithmically spaced in the interval $[10^{-3}, 0.9]$ and 80 eigenvalues uniformly spaced in the interval $[1, 2]$; we seek a few of the smallest eigenvalues. Figure 2.1 shows $\pi(z)$ for degree $d = 1, 2, 4$ and 8 (red lines) and the values of $\pi(\lambda_j)$ (black dots and gray lines). Since the small eigenvalues of A are near the origin (where $\pi(0) = 1$), $\pi(\lambda_j)$ is large for these values, while $\pi(\lambda_j)$ is small for the larger eigenvalues. Moreover, π *separates the tightly clustered eigenvalues near the origin.* The desired eigenvalues of $\pi(A)$ (nearest 1) will be easier for Arnoldi to compute than the corresponding (smallest) eigenvalues of A. Figure 2.1 also hints at a complication: when the degree is large, the map π entangles some of the larger eigenvalues from the interval $[10^{-3}, 0.9]$ with those from $[1, 2]$.

The GMRES polynomial π is easy to construct in factored form. To find its roots, run a cycle of GMRES(d) and compute the harmonic Ritz values [12, 20, 22, 24] (reciprocals of Rayleigh–Ritz eigenvalue estimates for A^{-1} from $A\mathcal{K}_d(A, b)$). For numerical stability, label the roots using the modified Leja ordering [1, alg. 3.1], giving

$$\pi(z) = (1 - z/\theta_1)\cdots(1 - z/\theta_d). \tag{2.1}$$

[2] This form allows one to write $\pi(z) = 1 - z\varphi(z)$ for some $\varphi \in \mathcal{P}_{d-1}$. Then $0 \approx \pi(A)b = (I - A\varphi(A))b$ suggests that $\varphi(A) \approx A^{-1}$. Thornquist uses this φ as the polynomial preconditioner, replacing A with $A - \mu B$ for generalized eigenvalue problems [38].

A quick listing of the algorithm follows.

Polynomial Preconditioned Arnoldi with GMRES polynomial of degree d

1. **Construction of the polynomial preconditioner, π:**
   (a) Run one cycle of GMRES(d).
   (b) Find the harmonic Ritz values, $\theta_1, \ldots, \theta_d$, the roots of the GMRES polynomial: given the Arnoldi decomposition $AV_d = V_{d+1}H_{d+1,d}$, find the eigenvalues of $H_{d,d} + h_{d+1,d}^2 f e_d^T$, where $f = H_d^{-*} e_d$ with elementary coordinate vector $e_d = [0, \ldots, 0, 1]^T$.
   (c) Order the GMRES roots with modified Leja ordering [1, alg. 3.1]. (For guidance on handling underflow or overflow, see [11, p. 4].)
2. **PP-Arnoldi:** Apply restarted Arnoldi to the matrix $\pi(A) = \Pi_{i=1}^{d}(I - A/\theta_i)$. (The experiments in Sections 4–8 use a thick-restarted Arnoldi method with exact shifts [21, 22, 35, 40].)

3. Convergence theory for polynomial preconditioning. Let $\sigma(A)$ denote the spectrum of A. We seek some subset $\Sigma := \{\lambda_1, \ldots, \lambda_k\}$ of $\sigma(A)$, typically those eigenvalues having smallest magnitude. The eigenvalues in Σ should be listed with their full multiplicity, but we presume that the eigenvalues in Σ are *nonderogatory* (i.e., there exists only one linearly independent eigenvector for each distinct eigenvalue in Σ); this assumption is standard for Krylov subspace convergence theory, required because of the concept of *reachable invariant subspaces* [3, 4].

3.1. Polynomial preconditioning in a simple setting. Polynomial preconditioning using the GMRES residual polynomial is a sophisticated algorithm that is challenging to analyze in fine detail. To indicate why it works, in this subsection we study the simple case of Hermitian A and degree $d = 2$ polynomial, before presenting more general but less detailed analysis in the subsequent subsections.

Suppose A is Hermitian with positive eigenvalues $\lambda_1 < \lambda_2 < \cdots < \lambda_n$, and we seek to compute λ_1 using the standard Arnoldi (Lanczos) method. We can bound convergence at the mth step using a polynomial $\phi \in \mathcal{P}_m$ satisfying $\phi(\lambda_1) = 1$, with magnitude as small as possible over $[\lambda_2, \lambda_n]$. Standard theory (e.g., [28, Thm. 2.2], [31, sect. 6.11], [32, chap. 6]) uses Chebyshev polynomials to bound convergence of the Ritz residual at the mth step by $C\rho^m$, with constant C and asymptotic rate[3]

$$\rho \equiv \frac{\sqrt{K} - 1}{\sqrt{K} + 1}, \qquad K \equiv \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1}. \tag{3.1}$$

Does polynomial preconditioning improve this asymptotic rate? Suppose we apply the Arnoldi algorithm to the matrix $\pi(A)$ for the quadratic GMRES polynomial

$$\pi(z) = \Big(1 - \frac{z}{\theta_1}\Big)\Big(1 - \frac{z}{\theta_2}\Big).$$

Here θ_1 and θ_2 are the harmonic Ritz values[4] from $\mathcal{K}_2(A, b)$ for some b, labeled

$$\lambda_1 \le \theta_1 \le \theta_2 \le \lambda_n.$$

[3] Here K behaves like the condition number of a matrix, and the convergence rate resembles that obtained for the conjugate gradient algorithm with the origin shifted to λ_1.

[4] Since harmonic Ritz values are reciprocals of standard Ritz values for A^{-1} (from the space $A\mathcal{K}_2(A, b)$) and A is Hermitian, $1/\theta_1$ and $1/\theta_2$ must fall between the extreme eigenvalues of A^{-1}, i.e., in $[1/\lambda_n, 1/\lambda_1]$. Thus their reciprocals, the harmonic Ritz values, are located in $[\lambda_1, \lambda_n]$. For more details, see [2]; the situation is more complicated for non-Hermitian problems.

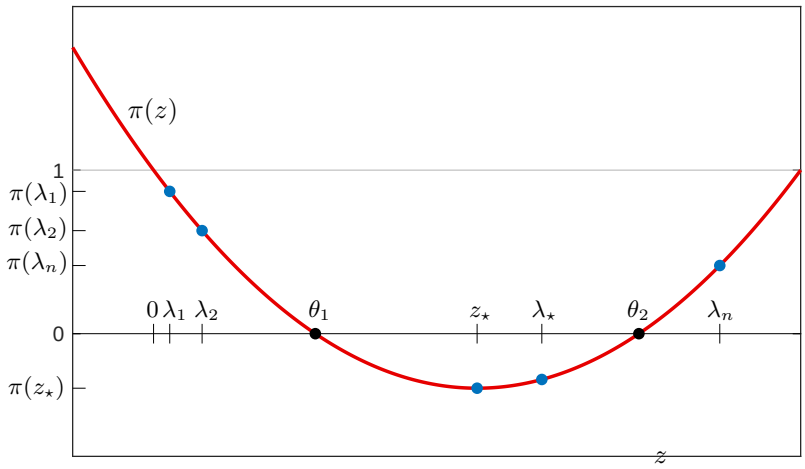

FIG. 3.1. *The quadratic polynomial $\pi(z)$, which attains its minimum at $z_\star \in [\lambda_1, \lambda_n]$.*

To show that polynomial preconditioning is effective, we must argue that $\pi(\lambda_1)$ is better separated from the rest of the spectrum of $\pi(A)$ than λ_1 was from the rest of the spectrum of A, i.e., we should show that Arnoldi finds $\pi(\lambda_1)$ with a faster convergence rate than the standard approach.

Figure 3.1 illustrates the setting. The location of the unwanted eigenvalues of $\pi(A)$ relative to $\pi(\lambda_1)$ determines the convergence rate. We must bound the unwanted eigenvalues in an interval that does not contain $\pi(\lambda_1)$.

- Complications arise if $\pi(\lambda_1) < \pi(\lambda_n)$, in which case the desired eigenvalue $\pi(\lambda_1)$ is in the interior of the spectrum of $\pi(A)$. We seek conditions that eliminate this possibility. In fact, we prefer the stronger condition $\pi(\lambda_2) \geq \pi(\lambda_n)$, to ensure $\pi(\lambda_2)$ is the eigenvalue of $\pi(A)$ closest to $\pi(\lambda_1)$. Simple algebra reduces $\pi(\lambda_2) \geq \pi(\lambda_n)$, that is,

$$\Big(1 - \frac{\lambda_2}{\theta_1}\Big)\Big(1 - \frac{\lambda_2}{\theta_2}\Big) \geq \Big(1 - \frac{\lambda_n}{\theta_1}\Big)\Big(1 - \frac{\lambda_n}{\theta_2}\Big),$$

  to the condition[5]

$$\theta_1 + \theta_2 \geq \lambda_2 + \lambda_n,$$

  suggesting a practical way to deter difficulties: compute degree-2 GMRES polynomials for a few b, and select the one that maximizes $\theta_1 + \theta_2$. (This can be done without knowledge of the eigenvalues of A.)
- To bound the spectrum of $\pi(A)$, we must know its most negative extent,

$$\min_{\lambda \in \{\lambda_2, \ldots, \lambda_n\}} \pi(\lambda).$$

  Suppose this minimum is attained for $\lambda = \lambda_\star$. For later use, we will also consider the global minimum of $\pi(z)$, which gives a lower bound on $\pi(\lambda)$ for $\lambda \in \sigma(A)$. Setting the derivative of π to zero yields the global minimizer

$$z_\star = \frac{\theta_1 + \theta_2}{2},$$

[5] If we merely have that $\theta_1 + \theta_2 > \lambda_1 + \lambda_n$, the method will still converge, but $\pi(\lambda_n)$ will determine the asymptotic rate, rather than $\pi(\lambda_2)$.

attaining the minimum

$$\pi(z_\star) = \frac{1}{2} - \frac{\theta_1^2 + \theta_2^2}{4\theta_1\theta_2}.$$

Provided $\theta_1 + \theta_2 \geq \lambda_2 + \lambda_n$, the unwanted eigenvalues of $\pi(A)$ fall in the interval

$$\big[\pi(\lambda_\star), \pi(\lambda_2)\big] \subseteq \big[\pi(z_\star), \pi(\lambda_2)\big].$$

The quadratic polynomial preconditioned Arnoldi convergence rate is thus

$$\widehat{\rho} := \frac{\sqrt{\widehat{K}} - 1}{\sqrt{\widehat{K}} + 1}, \qquad \widehat{K} \equiv \frac{\pi(\lambda_1) - \pi(\lambda_*)}{\pi(\lambda_1) - \pi(\lambda_2)} \leq \frac{\pi(\lambda_1) - \pi(z_*)}{\pi(\lambda_1) - \pi(\lambda_2)}.$$

Substituting the formula $\pi(z) = 1 - z(\theta_1 + \theta_2)/(\theta_1\theta_2) + z^2/(\theta_1\theta_2)$ into the expression for $\widehat{K}$ and simplifying leads to the appealing formula

$$\begin{aligned}
\widehat{K} &= \frac{(\theta_1 + \theta_2)(\lambda_\star - \lambda_1) + (\lambda_1^2 - \lambda_\star^2)}{(\theta_1 + \theta_2)(\lambda_2 - \lambda_1) + (\lambda_1^2 - \lambda_2^2)} \\
&= \left(\frac{\lambda_\star - \lambda_1}{\lambda_2 - \lambda_1}\right)\left(\frac{\theta_1 + \theta_2 - (\lambda_1 + \lambda_\star)}{\theta_1 + \theta_2 - (\lambda_1 + \lambda_2)}\right) \\
&= \left(\frac{\lambda_\star - \lambda_1}{\lambda_2 - \lambda_1}\right)\left(1 - \frac{\lambda_\star - \lambda_2}{\theta_1 + \theta_2 - (\lambda_1 + \lambda_2)}\right).
\end{aligned}$$

Compare this $\widehat{K}$ to the earlier factor K in (3.1) for the standard Arnoldi method without preconditioning:

$$K = \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1}.$$

Smaller values of K and $\widehat{K}$ give faster convergence rates ρ and $\widehat{\rho}$. Since $\lambda_* \leq \lambda_n$,

$$\frac{\lambda_\star - \lambda_1}{\lambda_2 - \lambda_1} \leq \frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1},$$

with equality holding only when $\lambda_\star = \lambda_n$. Since further

$$1 - \frac{\lambda_\star - \lambda_2}{\theta_1 + \theta_2 - (\lambda_1 + \lambda_2)} \leq 1,$$

with equality holding only when $\lambda_* = \lambda_2$, we can conclude that $\widehat{K} < K$: quadratic polynomial preconditioning will improve the convergence rate, under the mild condition $\theta_1 + \theta_2 \geq \lambda_2 + \lambda_n$. This improved rate comes at the cost of two matrix-vector products per iteration for the quadratically preconditioned case.

Is this improvement reflected in practice? Take a simple example with $A = \mathrm{diag}(1, 2, 3, \ldots, 1000)$, giving $K = 999$. The value of $\widehat{K}$ varies, but the average for 10 trials with random starting vectors b is much smaller, $\widehat{K} \approx 302$. Ten trials using the restarted Arnoldi method with quadratic polynomial preconditioning (with maximum subspace dimension 10, keeping 1 Ritz value at restart) to find $\lambda_1 = 1$ took an average of 851 matrix-vector products, compared to an average of 1298 matrix-vector products with no polynomial preconditioning: simple quadratic preconditioning gives a 34% reduction. The results of these 10 tests vary considerably, from 688 to 954 matrix-vector products with polynomial preconditioning and from 991 to 1481 without. However, $\theta_1 + \theta_2$ changes only between 1.1865 to 1.2217. The results seem more correlated to the magnitude of the first component in the starting vector.

3.2. Invariant subspace convergence theory. Now consider the general setting. How does the preconditioned Arnoldi algorithm converge as the dimension of the Krylov subspace increases? Especially for nonsymmetric A, the error in individual eigenvalue estimates does not always decrease at a consistent rate. Thus we prefer to analyze the rate at which the Krylov subspace "captures" the invariant subspace (span of eigenvectors and generalized eigenvectors) associated with $\Sigma \equiv \{\lambda_1, \ldots, \lambda_k\}$.

Let us make this notion precise. Let $\mathcal{U}$ denote the maximal invariant subspace associated with the desired eigenvalues Σ. Since none of these eigenvalues are derogatory, $\dim(\mathcal{U}) = k$. The Arnoldi algorithm approximates $\mathcal{U}$ by some Krylov subspace $\mathcal{V}$, whose dimension will generally exceed k. The *containment gap* (or just *gap*) between $\mathcal{U}$ and $\mathcal{V}$,

$$\delta(\mathcal{U}, \mathcal{V}) \equiv \max_{u \in \mathcal{U}} \min_{v \in \mathcal{V}} \frac{\|u - v\|}{\|u\|}, \tag{3.2}$$

measures the sine of the largest canonical angle between $\mathcal{U}$ and its best k-dimensional approximation from $\mathcal{V}$. Note that $\delta(\mathcal{U}, \mathcal{V}) \in [0, 1]$, with $\delta(\mathcal{U}, \mathcal{V}) = 1$ when $\dim(\mathcal{V}) < \dim(\mathcal{U}) = k$: $\mathcal{V}$ must have dimension at least k in order to approximate all of $\mathcal{U}$. For additional properties of the gap, see [3, 6, 14].

For the polynomially preconditioned Arnoldi algorithm, the Krylov subspace $\mathcal{K}_m(\pi(A), v)$ plays the role of $\mathcal{V}$. How does $\delta(\mathcal{U}, \mathcal{K}_m(\pi(A), v))$ depend on the choice of π, the dimension m, and the starting vector v? We shall apply the convergence theory from [4], which uses the following notation and assumptions.

(a) Label the desired eigenvalues as $\lambda_1, \ldots, \lambda_k$, allowing multiplicities. Assume none of these eigenvalues are derogatory, and these eigenvalues are disjoint from the rest of the spectrum $\{\lambda_{k+1}, \ldots, \lambda_n\}$.
(b) $\mathcal{U}$ denotes the k-dimensional invariant subspace associated with $\lambda_1, \ldots, \lambda_k$.
(c) P_g denotes the spectral projector onto $\mathcal{U}$, so that $P_b \equiv I - P_g$ is the spectral projector onto the invariant subspace associated with $\lambda_{k+1}, \ldots, \lambda_n$.
(d) Ω_b is a compact subset of $\mathbb{C}$ that contains the eigenvalues $\lambda_{k+1}, \ldots, \lambda_n$.
(e) $\alpha_g(z) \equiv (z - \lambda_1) \cdots (z - \lambda_k)$ is the component of the minimal polynomial of A associated with the desired eigenvalues.

For the sake of comparison, we start with Theorem 3.3 of [4] for the standard Arnoldi method applied to (A, v). For a Krylov subspace of dimension $m \geq 2k$,

$$\delta(\mathcal{U}, \mathcal{K}_m(A, v)) \leq \left(\max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(A) P_b v\|}{\|\psi(A) P_g v\|} \right) \kappa_A(\Omega_b) \min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b} |1 - \alpha_g(z) p(z)|. \tag{3.3}$$

This bound has three ingredients.

- The starting vector v only appears in the constant

$$C_1 \equiv \max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(A) P_b v\|}{\|\psi(A) P_g v\|}.$$

  If $k = 1$, $C_1 = \|P_b v\| / \|P_g v\|$ gauges the bias of v toward the desired invariant subspace. For $k > 1$, the eigenvalues of A also influence C_1. For additional details about this constant, see [3, sect. 5.1].
- The constant

$$C_2 \equiv \kappa_A(\Omega_b) \geq 1$$

is a measure of the nonnormality of A associated with the unwanted eigenvalues. If A is symmetric, then $\kappa_A(\Omega_b) = 1$. For nonnormal A, enlarging Ω_b to contain points beyond $\lambda_{k+1}, \ldots, \lambda_n$ typically reduces $\kappa_A(\Omega_b)$. For a detailed discussion of this constant, see [3, sect. 5.2].

- As the Krylov subspace dimension m grows, the approximation problem

$$\min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b} |1 - \alpha_g(z)p(z)| \tag{3.4}$$

gives the mechanism for convergence. The minimization problem seeks polynomials that approximate $1/\alpha_g(z) = (z - \lambda_1)^{-1} \cdots (z - \lambda_k)^{-1}$ over $z \in \Omega_b$. The convergence of this polynomial approximation problem depends on the proximity of Ω_b to the desired eigenvalues $\lambda_1, \ldots, \lambda_k$. Better separation between the desired and undesired eigenvalues yields faster convergence.

3.3. Convergence theory for polynomial preconditioning. Polynomial preconditioning alters the convergence bound (3.3), replacing A by $\pi(A)$. Note that if π maps two distinct eigenvalues $\lambda_j \neq \lambda_\ell$ to exactly the same point, $\pi(\lambda_j) = \pi(\lambda_\ell)$, then the theory of reachable invariant subspaces [4, sect. 2.1] dictates that the Krylov subspace $\mathcal{K}_m(\pi(A), v)$ cannot contain both eigenvectors associated with λ_j and λ_k (it will typically contain a one-dimensional subspace of their span). Thus, we must avoid the case where any eigenvalue $\pi(\lambda_1), \ldots, \pi(\lambda_k)$ is derogatory.

THEOREM 3.1. *Suppose $\pi(\lambda_1), \ldots, \pi(\lambda_k)$ are non-derogatory eigenvalues of $\pi(A)$, and $\{\pi(\lambda_1), \ldots, \pi(\lambda_k)\} \cap \{\pi(\lambda_{k+1}), \ldots, \pi(\lambda_n)\} = \emptyset$. Using the notation established above, the gap between the desired invariant subspace $\mathcal{U}$ (associated with the non-derogatory eigenvalues $\lambda_1, \ldots, \lambda_k$ of A) and the polynomially-preconditioned Krylov subspace $\mathcal{K}_m(\pi(A), v)$ for $m \geq 2k$ satisfies*

$$\delta(\mathcal{U}, \mathcal{K}_m(\pi(A), v)) \leq \left(\max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(\pi(A))P_b v\|}{\|\psi(\pi(A))P_g v\|} \right) \kappa_{\pi(A)}(\Omega_b^\pi) \min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z)p(z)|.$$

Here Ω_b^π is a compact subset of $\mathbb{C}$ that contains $\pi(\lambda_{k+1}), \ldots, \pi(\lambda_n)$, and

$$\alpha_g^\pi(z) := \big(z - \pi(\lambda_1)\big) \cdots \big(z - \pi(\lambda_k)\big).$$

For this bound to converge, π must map the desired eigenvalues outside Ω_b^π:

$$\{\pi(\lambda_j)\}_{j=1}^k \ \cap \ \Omega_b^\pi \ = \ \emptyset.$$

Since the spectral projectors are invariant under the transformation $A \mapsto \pi(A)$ (provided the desired and undesired eigenvalues remain disjoint under π), P_b and P_g are the same for A and $\pi(A)$.

Consider the simplest case: seeing $k = 1$ eigenvalue of a symmetric A. Since $k = 1$, the constants in (3.3) and Theorem 3.1 that involve v are just $C_1 = \|P_b v\|/\|P_g v\|$. Since A is symmetric, $C_2 = \kappa_A(\Omega_b) = \kappa_{\pi(A)}(\Omega_b^\pi) = 1$. The only difference in the convergence bounds in (3.3) and Theorem 3.1 then comes from the polynomial approximation problems. As suggested by Figures 2.1 and 3.1, the map π can effectively separate the desired eigenvalues from the rest of the spectrum, making it possible that

$$\min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z)p(z)| \ \ll \ \min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b} |1 - \alpha_g(z)p(z)|.$$

Keep A symmetric but allow general $k \geq 1$. Presuming that π has real coefficients (ensuring $\pi(\sigma(A)) \subset \mathbb{R}$), denote

$$\Omega_b^\pi \equiv \Big[\min_{k+1 \leq j \leq n} \pi(\lambda_j), \ \max_{k+1 \leq j \leq n} \pi(\lambda_j) \Big] \subset \mathbb{R}.$$

Let $\lambda_* \in \{\lambda_1, \ldots, \lambda_k\}$ denote the desired eigenvalue that is mapped closest to Ω_b^π:

$$\operatorname{dist}(\pi(\lambda_*), \Omega_b^\pi) = \min_{1 \leq j \leq k} \operatorname{dist}(\pi(\lambda_j), \Omega_b^\pi) = \min_{1 \leq j \leq k} \ \min_{z \in \Omega_b^\pi} |\pi(\lambda_j) - z|.$$

Supposing that $\pi(\lambda_*) \notin \Omega_b^\pi$, define

$$K \equiv \frac{\max_{z \in \Omega_b^\pi} \ |\pi(\lambda_*) - z|}{\min_{z \in \Omega_b^\pi} \ |\pi(\lambda_*) - z|} \geq 1. \tag{3.5}$$

Then using standard Chebyshev approximation theory [31, sect. 6.11] (cf. (3.1)), the polynomial approximation problem in Theorem 3.1 converges at the asymptotic rate

$$\rho \equiv \frac{\sqrt{K} - 1}{\sqrt{K} + 1} \in [0, 1), \tag{3.6}$$

meaning that there exists some constant $C > 0$ such that for all $m \geq 2k$,

$$\min_{p \in \mathcal{P}_{m-2k}} \max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z) p(z)| \ \leq \ C \rho^m.$$

Table 3.1 compares this convergence rate ρ for the standard Arnoldi method (first row, $\pi(z) = z$) to the convergence obtained for minimum residual polynomials π of increasing degree d for the symmetric A used in Figure 2.1. We seek the $k = 5$ smallest magnitude eigenvalues of A. (When $d = 1$, the shift-invariance property of the Krylov subspace implies that polynomial preconditioning and standard Arnoldi generate the same approximation subspace: $\mathcal{K}_m(\pi(A), v) = \mathcal{K}_m(A, v)$.) Note that the fastest convergence rate *per polynomial degree*, $\rho^{1/d}$, is obtained for the standard case (or $d = 1$). This is expected: the preconditioned space $\mathcal{K}_m(\pi(A), v)$ only contains a small part of the much larger space $\mathcal{K}_{d(m-1)+1}(A, v)$. However, *the preconditioned method achieves its convergence rate while using subspaces of much lower dimension.*

3.4. The effect of restarting on convergence. To obtain accurate eigenvalue estimates while limiting the dimension m of the Krylov subspace (1.1), the Arnoldi algorithm is *restarted* [28, 34]. We begin by describing the standard restarted Arnoldi algorithm with no preconditioning. After taking m steps of the Arnoldi process with the matrix A and starting vector $v_0 \equiv v$ (the first *cycle*), the method runs a fresh set of m steps with the same A but a new starting vector, v_1. In general, cycle $c + 1$ takes m Arnoldi steps with A and starting vector v_c. These new starting vectors are formed using polynomial restart methods [28, 29, 34]: given some positive integer $r \leq m - k$, such methods construct $v_c = \phi_c(A) v_{c-1}$, where $\phi_c \in \mathcal{P}_r$. Aggregate these starting vectors to get $v_c = \Phi_c(A) v$, for $\Phi_c \equiv \phi_1 \cdots \phi_c \in \mathcal{P}_{cr}$, so that cycle $c + 1$ of the restarted Arnoldi method uses the Krylov subspace

$$\mathcal{K}_m(A, \Phi_c(A) v) = \operatorname{span}\{\Phi_c(A) v, A \Phi_c(A) v, \ldots, A^{m-1} \Phi_c(A) v\}, \tag{3.7}$$

generally an m-dimensional subspace of $\mathcal{K}_{cr+m}(A, v)$. For any $x \in \mathcal{K}_m(A, \Phi_c(A) v)$ there exists some $\omega \in \mathcal{P}_{m-1}$ such that

$$x = \omega(A) \Phi_c(A) v, \tag{3.8}$$

TABLE 3.1

Asymptotic convergence rates for various polynomial degrees d, for the example in Figure 2.1 with $k = 5$ desired eigenvalues. The first row is for standard Arnoldi, $\pi(A) = A$. The set Ω_g^π is the smallest interval that contains $\{\pi(\lambda_j)\}_{j=1}^k$, while Ω_b^π is the smallest interval that contains the map of the unwanted eigenvalues, $\{\pi(\lambda_j)\}_{j=k+1}^n$. When Ω_b^π and Ω_g^π overlap, we set $\rho = 1$.

d	Ω_g^π	Ω_b^π	ρ	$\rho^{1/d}$
standard	$[0.0010, 0.0042]$	$[\ 0.0060, 2.0000]$	0.9416	
1	$[0.9973, 0.9993]$	$[-0.3055, 0.9961]$	0.9416	0.9416
2	$[0.9936, 0.9985]$	$[-0.0705, 0.9908]$	0.9030	0.9503
3	$[0.9840, 0.9962]$	$[-0.2018, 0.9772]$	0.8589	0.9506
4	$[0.9690, 0.9925]$	$[-0.4384, 0.9557]$	0.8232	0.9525
5	$[0.9529, 0.9886]$	$[-0.4155, 0.9329]$	0.7845	0.9526
6	$[0.9292, 0.9829]$	$[-0.4102, 0.8994]$	0.7407	0.9512
7	$[0.9063, 0.9772]$	$[-0.4036, 0.8673]$	0.7057	0.9514
8	$[0.8753, 0.9695]$	$[-0.3909, 0.8240]$	0.6650	0.9503
16	$[0.5563, 0.8830]$	$[-0.3501, 0.4002]$	0.4134	0.9463
24	$[0.1046, 0.7217]$	$[-0.3096, 0.2434]$	1.0000	1.0000

with $\omega \cdot \Phi_c \in \mathcal{P}_{cr+m-1}$. Like polynomial preconditioning, restarting with polynomial filters gives access to elements in a high-degree Krylov subspace, while keeping the overall subspace dimension low.

The convergence behavior will depend on the polynomial filters [3, 4]. While these filters can be constructed from Chebyshev polynomials [29], filters built from "exact shifts" (unwanted Ritz values) [34] are simpler and more widely used. Such shifts always give convergence for symmetric A [34] (aside from pathological v_0), and usually work well for nonsymmetric A (though they can fail, in theory [9, 10]). Moreover, these filters can be implemented stably using the implicitly restarted Arnoldi [34], thick-restart Arnoldi [21, 22, 35, 40] and Krylov–Schur [36] algorithms. Since these exact-shift filters are built up during each cycle of the restarted Arnoldi procedure, they require numerous inner product evaluations.

Practical Arnoldi eigenvalue computations that use polynomial preconditioning will also incorporate polynomial restarting, using the approximation space

$$\mathcal{K}_m(\pi(A), \Phi_c(\pi(A))v) = \\ \operatorname{span}\{\Phi_c(\pi(A))v, \pi(A)\Phi_c(\pi(A))v, \ldots, \pi(A)^{m-1}\Phi_c(\pi(A))v\}. \tag{3.9}$$

Generally (3.9) is an m-dimensional subspace of $\mathcal{K}_{d(cr+m-1)+1}(A, v)$. Now for any $x \in \mathcal{K}_m(\pi(A), \Phi_c(\pi(A))v)$ there exists a polynomial $\omega \in \mathcal{P}_{m-1}$ such that

$$x = \omega(\pi(A))\Phi_c(\pi(A))v, \tag{3.10}$$

so preconditioning gives access to vectors x that can be written in terms of a polynomial $((\omega \circ \pi) \cdot (\Phi_c \circ \pi))$ of degree $d(cr + m - 1)$, i.e., d times larger than available with restarting alone in (3.8).

How do preconditioning and restarting combine to affect convergence? We first address this question by continuing the experiment started in Figure 2.1, seeking the $k = 5$ smallest magnitude eigenvalues. Figure 3.2 compares convergence of the polynomially-preconditioned Arnoldi algorithm, with and without restarting. In the top figures (no restarts), increasing the preconditioning polynomial degree d improves the convergence, but increases the number of matrix-vector products involving A.

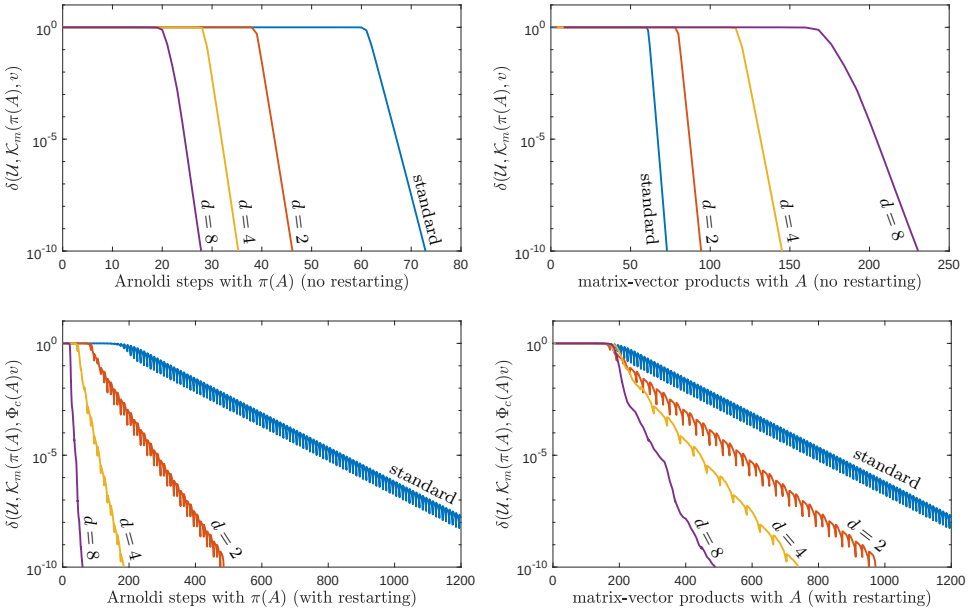

FIG. 3.2. *Arnoldi iterations and matrix-vector products for various choices of d and $k = 5$ eigenvalues, for the example in Figure 2.1 without restarts (top) and with restarts using $m = 2k$ (bottom). Without restarts, standard Arnoldi requires more steps (top left) but fewer matrix-vector products (top right). Restarting can give polynomial preconditioning another advantage, making it faster in terms of both steps (bottom left) and matrix-vector products (bottom right).*

The bottom figures incorporate polynomial restarting, limiting the Krylov subspace to have dimension $m = 2k$ and using filter polynomials of degree $r = k$ at each cycle. Polynomial preconditioning improves convergence by a larger margin, *enough to now deliver convergence using fewer matrix-vector products.*

By adapting [4, eq. (3.10)], we can also provide a bound on the gap between the desired invariant subspace $\mathcal{U}$ and the restarted Krylov subspace using polynomial preconditioning. Suppose the aggregate polynomial filter $\Phi_c \in \mathcal{P}_{cr}$ has R distinct roots (the shifts), also distinct from the images of the desired eigenvalues, $\Sigma^\pi = \{\pi(\lambda_1), \ldots, \pi(\lambda_k)\}$. Let $\Psi_c \in \mathcal{P}_{R-1}$ interpolate $1/\alpha_g^\pi$ at these R points. Then

$$\begin{aligned}&\delta(\mathcal{U}, \mathcal{K}_m(\pi(A), \Phi_c(A)v)) \\ &\qquad \le \left(\max_{\psi \in \mathcal{P}_{k-1}} \frac{\|\psi(\pi(A))P_b v\|}{\|\psi(\pi(A))P_g v\|} \right) \kappa_{\pi(A)}(\Omega_b^\pi) \max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z)\Psi_c(z)|. \quad (3.11)\end{aligned}$$

If the roots of Ψ_c are distributed throughout Ω_b^π, we expect $\max_{z \in \Omega_b^\pi} |1 - \alpha_g^\pi(z)\Psi_c(z)|$ to be small. Even for standard Arnoldi computations with nonsymmetric A, characterizing the precise locations of the shifts has proved extremely challenging [5, 9, 10]; still, the bound (3.11) indicates how the polynomial preconditioner and the shifts can combine to enhance convergence.

4. Experiments. How does polynomial preconditioning perform in practice? In this section we explore two examples involving nonsymmetric A. Here and in future sections, our experiments use the thick restarted Arnoldi method [22, 40], which we refer to as Arnoldi(m, k): m denotes the largest subspace dimension and $k < m$ denotes

the number of Ritz values kept at each restart.[6] We seek the $nev < k$ smallest magnitude eigenvalues, leaving a buffer of $k - nev$ eigenvalues to accelerate convergence. Each orthogonalization step is followed by a pass of reorthogonalization. Convergence is tested at each Arnoldi cycle using the original matrix A: Let $\nu_1, \ldots, \nu_m$ denote the Ritz values for $\pi(A)$ at the end of a cycle, ordered by increasing distance from 1 ($|1 - \nu_1| \le |1 - \nu_2| \le \cdots \le |1 - \nu_m|$), and let $y_1, \ldots, y_m \in \mathbb{C}^n$ denote the associated unit-norm Ritz vectors. Then the Rayleigh quotient $\mu_j \equiv y_j^* A y_j$ gives an eigenvalue estimate for A. When seeking $nev < k$ eigenvalues, we require $\|Ay_j - \mu_j y_j\| \le rtol$ for $j = 1, \ldots, nev$, with $rtol = \|A\| 10^{-8}$ unless otherwise stated. The reported operation counts give a rough impression of the matrix-vector products with A, dot products, and other vector operations (scalar multiplications and additions); these counts will vary a bit with implementation details (e.g., reorthogonalization strategy, which residuals are checked at each cycle, etc.). In each case, to explore robustness we use a random starting vector b to generate π, and a different random starting vector v for the Arnoldi iterations. (In practice one might naturally use the same starting vector to generate π and for the Arnoldi iterations.) We average our results over ten trials, to minimize variation due to these starting vectors.

Example 1. Consider a finite difference discretization of a convection-diffusion equation on the unit square with homogeneous Dirichlet boundary conditions. On the bottom half of the square, the operator is $-u_{xx} - u_{yy} + 20u_x$; on the top, $-100u_{xx} - 100u_{yy} + 2000u_x$. We use five increasingly fine discretizations that give matrices of size $n = 2500$, 10,000, 40,000, 160,000 and 640,000. These matrices are nonsymmetric, but have real spectra; they exhibit a significant departure from normality: the condition number of the eigenvector matrix computed by MATLAB is approximately 2.62×10^4 for $n = 2500$, and 3.73×10^4 for $n = 10{,}000$; for all these n, the Henrici number $\|A^T A - AA^T\|/\|A\|^2 \approx 0.095$. We seek the $nev = 15$ smallest magnitude eigenvalues and the corresponding eigenvectors using Arnoldi(50,20), meaning the maximum subspace dimension is $m = 50$, and $k = 20$ Ritz vectors are saved at the restart.

Figure 4.1 compares regular Arnoldi(50,20) to degree $d = 25$ polynomial preconditioned Arnoldi(50,20) for the five matrices, giving both the number of matrix-vector products for convergence (left axis) and an estimate of the total cost (right axis). For $n = 2{,}500$, polynomial preconditioning uses slightly more matrix-vector products. As the matrix size increases and the eigenvalue problem becomes more difficult, polynomial preconditioning becomes increasingly better in comparison. For $n = 640{,}000$, regular Arnoldi averages 207755 matrix-vector products, while polynomial preconditioned Arnoldi with $d = 25$ only averages 63,017.

The computational cost is estimated as $cost = nnzr \times mvps + vops$, where $nnzr \approx 5$ is the average number of nonzeros per row in A, *mvps* is the number of matrix-vector products, and *vops* is the number of length-n vector operations, such as dot products and daxpy's. Of course, the cost of an *mvp* compared to a *vop* depends on the computer and implementation, but this rough estimate shows the potential for polynomial preconditioning to reduce computational cost. In Figure 4.1, *cost* is associated with the right axis (asterisks for regular Arnoldi, squares for $d = 25$). The axes are scaled so the values on the left axis are one-fifth of the corresponding height on the right axis, allowing one to see what portion of *cost* is due to *mvps*. For regular Arnoldi, most matrix-vector products are accompanied by an orthogonalization step; preconditioning uses more matrix-vector products (to compute $\pi(A)v$) relative to vector

[6] All examples involve real matrices; to preserve real arithmetic during the restarting process, sometimes k is temporarily reduced to $k - 1$ to avoid splitting a conjugate pair of Ritz values.

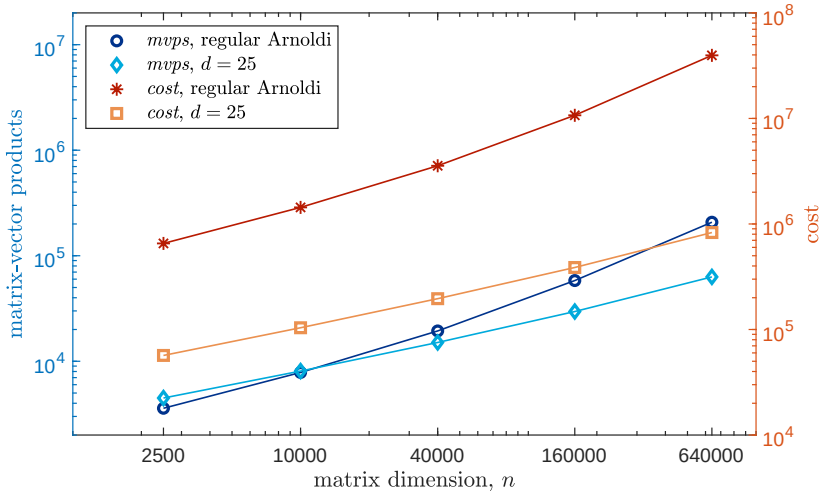

FIG. 4.1. *Example 1 (convection diffusion matrix): Comparison of regular Arnoldi(50,20) to polynomial preconditioned Arnoldi(50,20) with degree d = 25 for matrices of size 2,500, 10,000, 40,000, 160,000 and 640,000, averaged over 10 trials. Circles (regular Arnoldi) and diamonds (polynomial preconditioned) indicate the number of matrix-vector products (left axis). Asterisks and squares show the corresponding approximate cost (cost = nnzr×mvps+vops, right axis). (Polynomial preconditioning found the 15 leftmost eigenvalues in all trials; one trial of the unpreconditioned version for n = 10,000 missed the 14th eigenvalue.)*

operations. For this sparse A, *vops* dominate *mvps* for regular Arnoldi. Polynomial preconditioning with $d = 25$ reduces the *vops* per matrix-vector product by a factor of about 22 or 23 for all the n values shown here. With $n = 2{,}500$, the *cost* for regular Arnoldi is 654,079.52, but only 56,785.66 for polynomial preconditioning. With $n = 640{,}000$, the comparison is 39,614,862.23 to 829,431.11. Increasing the degree to $d = 100$ drops the cost to 523,878.07 : *over 75 times cheaper than regular Arnoldi.*

We continue the example by looking at the polynomials for the matrix of size $n = 10{,}000$. The upper portion of Figure 4.2 shows representative GMRES polynomials of degree 10 and 25, evaluated at all of the eigenvalues. The steeper slope at the origin of the $d = 25$ polynomial better separates the small eigenvalues from the others, as is clear from the close-up plot on the bottom. The desired first $nev = 15$ eigenvalues come first, followed by the next $k - nev = 5$ eigenvalues that serve as a buffer for the desired ones, and finally the next few eigenvalues: the polynomial mapping separates the small eigenvalues from the others.

A gap ratio for the 15th eigenvalue that takes into account the buffer Ritz values is $\frac{\lambda_{21}-\lambda_{15}}{\lambda_{10000}-\lambda_{21}} = 2.00 \times 10^{-5}$ for the matrix A, which gives some indication of the eventual convergence of that eigenvalue. With $d = 10$ polynomial preconditioning, the gap ratio improves to $\frac{\pi(\lambda_{21})-\pi(\lambda_{15})}{\pi(\lambda_{5100})-\pi(\lambda_{21})} = 1.46 \times 10^{-3}$, and for $d = 25$, it is better still: $\frac{\pi(\lambda_{21})-\pi(\lambda_{15})}{\pi(\lambda_{5146})-\pi(\lambda_{21})} = 8.48 \times 10^{-3}$. While these ratios suggest that polynomial preconditioning improves the convergence rate in terms of Arnoldi cycles, a larger gap ratio *does not* guarantee a reduction of overall matrix-vector products, since each iteration with a higher degree polynomial requires more of them. As Figure 4.1 shows, for $n = 10{,}000$ the number of matrix-vector products for $d = 25$ is a bit smaller than for regular Arnoldi.

Figure 4.3 shows results for $n = 160{,}000$ using different degree polynomials. Standard Arnoldi(50,20) is plotted at $d = 0$; then polynomial preconditioned Arnoldi(50,20) is applied with $d = 5, 10, 15, \ldots, 50$. The matrix-vector products, denoted on the left

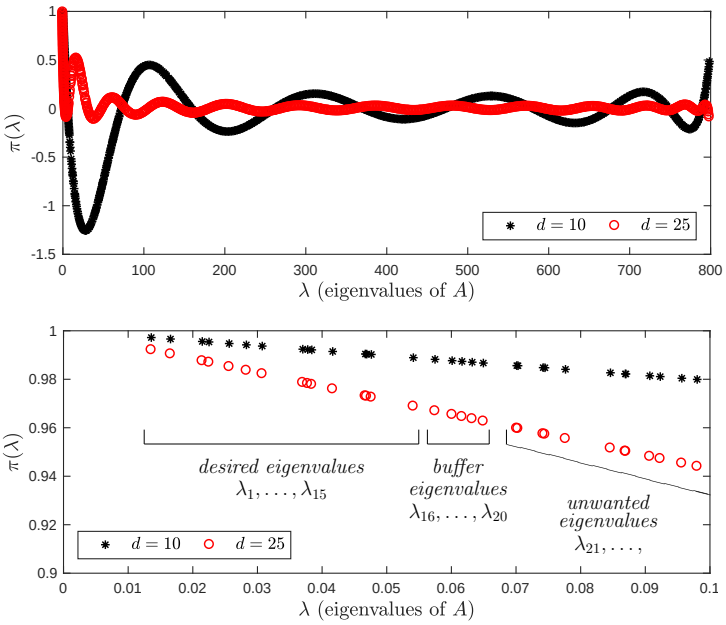

FIG. 4.2. *Example 1 (convection diffusion matrix, $n = 10{,}000$): Polynomials of degree $d = 10$ (black) and $d = 25$ (red) for the convection-diffusion matrix of size $n = 10{,}000$. The upper plot shows the polynomial values $\pi(\lambda)$ plotted for all eigenvalues λ of A. The bottom plot zooms in on the smallest eigenvalues, highlighting the sought-after $nev = 15$ smallest magnitude eigenvalues, the buffer of $k - nev = 5$ additional eigenvalues, and a few of the remaining unwanted eigenvalues.*

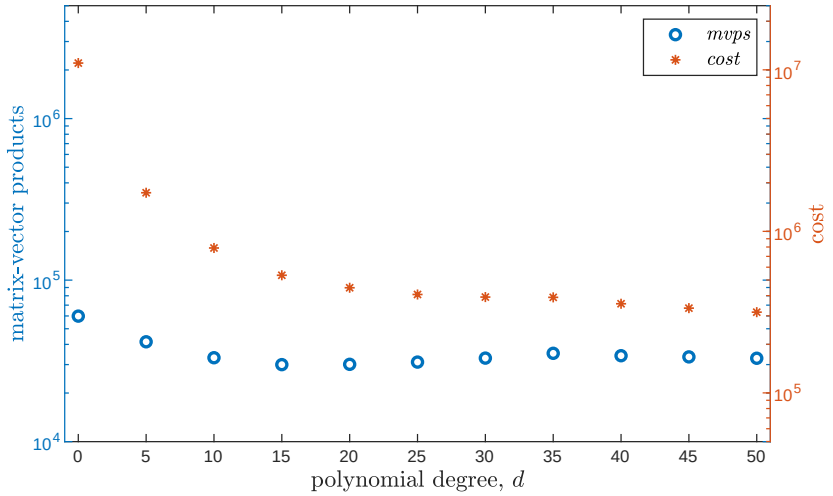

FIG. 4.3. *Example 1 (convection diffusion matrix, $n = 160{,}000$): A comparison of convergence between standard Arnoldi(50,20) ($d = 0$) and polynomial preconditioned Arnoldi(50,20) with $d = 5, 10, 15, \ldots, 50$. Circles indicate the number of matrix-vector products (left axis). The approximate cost ($cost = nnzr \times mvps + vops$) is indicated with asterisks (right axis).*

axis, hit a minimum for $d = 15$ and increase slightly beyond that. The cost estimate (right axis) decreases. For $d = 50$, $cost \approx 316{,}060$. The *cost* can go down a bit further with higher d; for $d = 150$, $cost \approx 298{,}222$. See Table 8.1 for further testing with a larger version of this example, including timings and dot product counts.

Example 2. The matrix E20r0100 from SuiteSparse [7] has a complex spectrum and a moderate departure from normality. (The condition number of the matrix of eigenvectors computed by MATLAB is approximately 1.67×10^6.) The matrix has dimension $n = 4241$ and an average of nearly 31 nonzeros per row. As before, we seek the $nev = 15$ eigenvalues nearest the origin, which are in the interior of the spectrum (A has 1199 eigenvalues with (quite small) negative real parts). Here we use Arnoldi(100,30) to access larger subspaces. Figure 4.4 shows how polynomial preconditioning changes the spectrum. The top left shows the spectrum of A, with a close-up on the right of the smallest-in-magnitude eigenvalues. The middle left portion of Figure 4.4 shows the resulting spectrum of $\pi(A)$ for a preconditioner of degree $d = 10$. We expect π to map the eigenvalues of A near the origin close to 1; this is the case, but these eigenvalues remain in the interior of the spectrum. They are shown in the middle right portion of the figure. Similarly, the degree 25 spectrum is shown in the bottom portions of the figure. Note that even with just $d = 10$, the spectrum improves: many eigenvalues are mapped near the origin and the eigenvalue originally nearest the origin is relatively better separated from its nearest neighbor by a factor of about 40 after the mapping. Table 4.1 gives computational results for different preconditioner degrees. A degree $d = 15$ polynomial reduces the number of matrix-vector products by a factor of almost 9 and vector operations by a factor of more than 125. Matrix-vector products are not further reduced with higher degree polynomials, but the vector operations are. Even low degree polynomials are effective for this example, but this is not always true for a tougher complex spectrum; see Example 2 with matrix Af23560 in [11, p. 18].

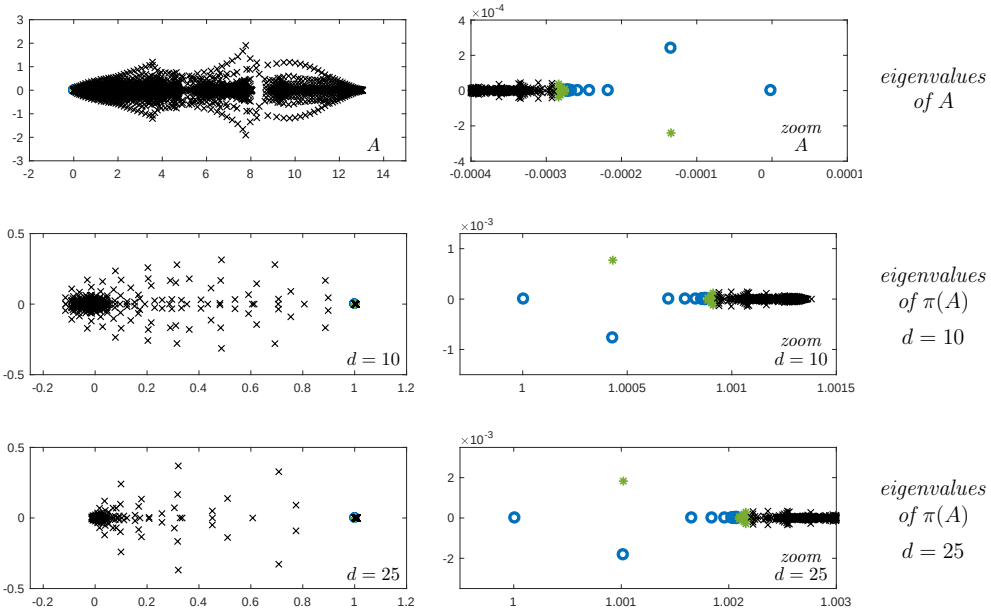

FIG. 4.4. *Example 2 (matrix E20r0100). The top plots show the eigenvalues of A, with the $nev = 15$ desired eigenvalues (nearest 0) as blue circles and the $k - nev = 15$ buffer eigenvalues as green stars; the undesired eigenvalues are shown as black $\times$ symbols. The middle plots show the eigenvalues of the preconditioned matrix $\pi(A)$ for degree $d = 10$; the desired eigenvalues are mapped near 1. The bottom plots show the eigenvalues of $\pi(A)$ for $d = 25$.*

TABLE 4.1
Example 2 (Matrix E20r0100, $n = 4241$): results for polynomial preconditioning with different degree polynomials using Arnoldi(100,30) to seek $nev = 15$ eigenvalues, averaged over 10 trials.

degree d	cycles	*mvps*	*vops* (millions)	dot products (millions)	*cost* (millions)	# correct eigenvalues
0	7,959.1	558,258.0	171.80	74.41	189.10	14.8
5	308.6	109,067.6	6.82	2.89	10.20	14.0
10	97.4	68,926.6	2.19	0.91	4.33	14.0
15	58.9	62,740.8	1.35	0.55	3.29	14.0
20	46.0	65,510.1	1.07	0.43	3.10	14.0
25	43.2	76,936.4	1.02	0.41	3.41	14.0
50	44.5	158,253.2	1.14	0.42	6.04	14.7
75	27.9	149,730.0	0.77	0.27	5.41	14.8
100	14.8	107,274.7	0.46	0.15	3.78	14.0

5. Two starting vectors. For the examples in the last section we randomly generated the starting vector b used to create π, an approach that works quite well in our experience. However, one might wonder: What happens if one makes a poor choice for b? How can one hedge against such a case? Here we consider the possibility that an unusual or skewed starting vector for π can give bad results, and show how two starting vectors can be used to generate π to minimize such risk.

Example 3. Let A be diagonal with $1, 2, 3, \ldots, 1000$ on the main diagonal. We run Arnoldi(50,20) to calculate $nev = 15$ eigenvalues to residual norm of 10^{-8} and use a different starting vector v for the polynomial preconditioned Arnoldi loop than the vector b used to develop π (with $d = 10$). The method requires only one cycle to find all 15 correct eigenpairs. Next, we make the last 100 components of the starting vector b for the polynomial small by multiplying them by 0.01. The resulting π is not small much past $\lambda = 930$; $\pi(\lambda)$ goes up to at least 7 at $\lambda = 1000$, transforming the problem of finding the eigenvalues near 1 into an interior eigenvalue problem. Convergence is much slower, with 16.8 cycles needed (average of 10 trials), and then only the first four desired eigenvalues ($\lambda = 1, 2, 3, 4$) are found. The remaining computed eigenvalues fall in $\{938, 939, \ldots, 956\}$, depending on the run: eigenvalues λ that π maps closer to the target point 1 than the desired eigenvalues $\lambda = 5, \ldots, 15$.

To mitigate against the risk of choosing a bad b vector, we propose an algorithm that uses two starting vectors to determine one polynomial, applying GMRES to a 2×2 block diagonal system of dimension $2n$.

Polynomial Determined by Two Starting Vectors

1. **Set-up:** Generate random vectors b_1 and b_2, with $\|b_1\| = \|b_2\| = 1/\sqrt{2}$. Let
$$\widehat{b} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \qquad \widehat{A} = \begin{bmatrix} A & 0 \\ 0 & A \end{bmatrix}.$$
2. **Generate polynomial:** Run a cycle of GMRES(d) with starting vector $\widehat{b}$ and matrix $\widehat{A}$, and find the roots of the GMRES polynomial π.

This approach essentially uses two Krylov subspaces, one each with b_1 and b_2, and so takes into account both starting vectors. We tested Example 5 with the skewed starting vector for b_1, but b_2 a random vector. The results are good: we now need only one or two cycles to compute all the desired small eigenvalues.

6. Damped polynomials. In certain situations, the GMRES polynomial maps too many (i.e., more than *nev*) small eigenvalues λ to large values of $|\pi(\lambda)|$. This concern can be reduced through the use of *damped polynomials*.

6.1. Overenthusiastic polynomials. The next example motivates the utility of a damped GMRES polynomial; it also illustrates that polynomial preconditioning can be effective even for a not especially sparse matrix.

Example 4. Consider S1rmq4m1 from SuiteSparse, a symmetric, positive definite matrix of size $n = 5489$ with an average of $nnzr \approx 47.8$ nonzeros per row. Finding the small eigenvalues is difficult because they are packed close together relative to the whole spectrum. The first 15 range from 0.3797 to 1.9027, the 21st is 2.2630 and the largest is 6.87×10^5. Figure 6.1 shows the performance of Arnoldi(50,20) applied to find the $nev = 15$ smallest magnitude eigenvalues of A. The gap ratio $\frac{|\lambda_{21}-\lambda_{15}|}{|\lambda_{5489}-\lambda_{21}|} \approx 5.24 \times 10^{-7}$ roughly describes the eventual convergence for the 15th eigenvalue. For typical preconditioning with $d = 20$, the gap ratio improves three orders of magnitude to $\frac{|\pi(\lambda_{21})-\pi(\lambda_{15})|}{|\pi(\lambda_{2737})-\pi(\lambda_{21})|} = \frac{|0.99728-0.99771|}{|-1.82056-0.99728|} \approx 1.54 \times 10^{-4}$, though recall each preconditioned Arnoldi iteration requires 20 matrix-vector products. Nevertheless, there is an improvement in matrix-vector products by a factor of 6.5 over these ten trials, and $cost = nnzr \times mvps + vops$ is reduced by a factor of 25.51. Though A has a relatively large number of nonzeros, $nnzr \approx 47.8$, the cost of vector operations still dominates for regular Arnoldi; in contrast, with polynomial preconditioning the matrix-vector products are the bigger expense. Moreover, for $d = 20$ polynomial preconditioning decreases the dot products by a factor of 130.34 over regular Arnoldi.

Figure 6.1 shows that polynomials of degree $d > 20$ can cause problems, giving much less consistent performance across the ten trials; for each $d > 20$, at least one run failed to find the correct $nev = 15$ smallest eigenvalues. What is going on here? Figure 6.2 indicates the problem, showing $\pi(\lambda)$ for $d = 10$, 20, and 30 for the first of our ten trials. The slope of π at the origin is much steeper for $d = 20$ than for $d = 10$,

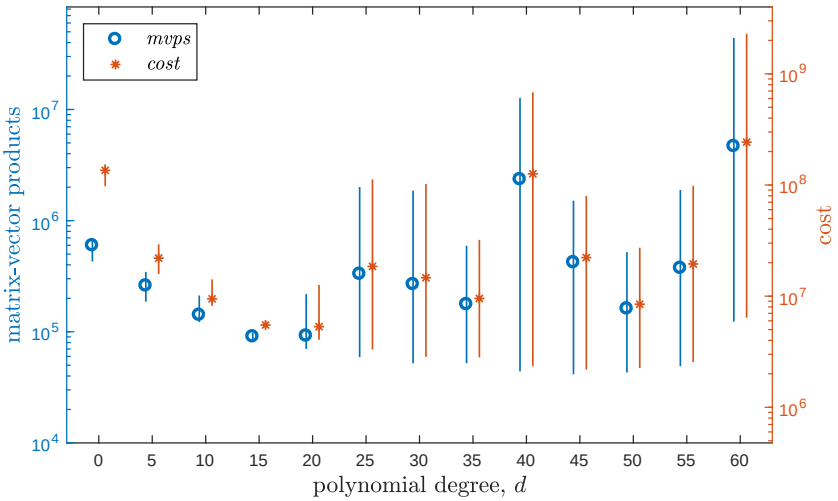

FIG. 6.1. *Example 4 (S1rmq4m1 matrix): Convergence of standard Arnoldi(50,20) ($d = 0$) and polynomial preconditioned Arnoldi(50,20) with $d = 5, 10, 15, \ldots, 60$. Blue circles indicate the average number of matrix-vector products (left axis); red asterisks show the average approximate cost ($cost = nnzr \times mvps + vops$, right axis). These results vary widely over ten trials; vertical bars show the minimum and maximum matrix-vector products and cost over these trials.*

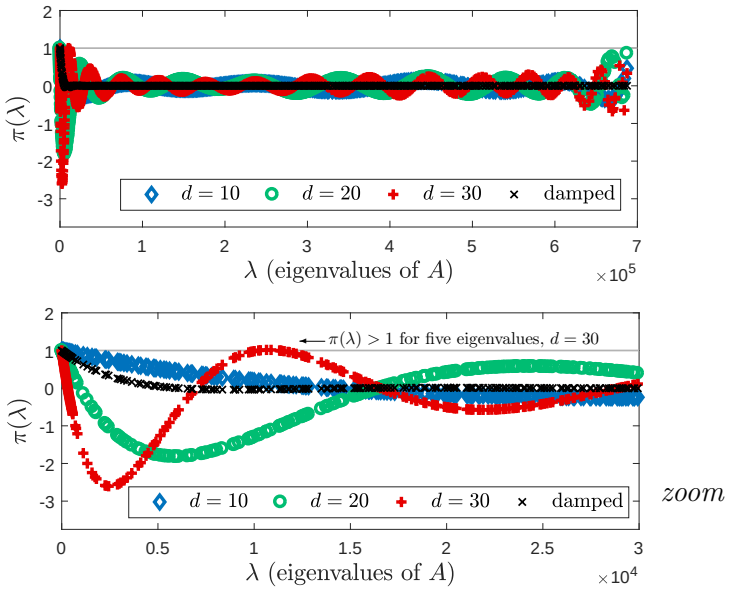

FIG. 6.2. *Example 4 (S1rmq4m1 matrix): Polynomials of degree $d = 10$, 20, 30, then the damped polynomial for $d = 30$. The top plot shows $\pi(\lambda)$ for all eigenvalues λ of A; the bottom plot zooms in near the origin. The damped polynomial is much better at isolating small magnitude eigenvalues of A as largest eigenvalues of $\pi(A)$.*

explaining the faster convergence. Degree $d = 30$ is even steeper, but has a problem near $\lambda = 10{,}000$, where $\pi(\lambda) > 1$ for five eigenvalues: π maps eigenvalues of A from the interior of the spectrum to the exterior of $\pi(A)$, mixing seven of them amongst or above the $nev = 15$ desired smallest eigenvalues of $\pi(A)$ near 1. The resulting interior eigenvalue problem can lead to slow convergence and spurious eigenvalues. Of the converged Ritz values for this example, only 11 fall among the *nev* desired smallest magnitude eigenvalues; the others are unwanted interior eigenvalues of A. As Figure 6.1 shows, erratic convergence continues for larger values of d.

We call the polynomials that jump up too high "overenthusiastic." This matrix seems prone to such polynomials because about half its spectrum is near the origin (2703 eigenvalues are less than 600; the next 2786 go from 700 up to 6.87×10^5): the GMRES polynomial has more roots near these small eigenvalues, with roots located less precisely amongst the others. One might caution users to stick with smaller degree d for problems like this, but we suggest an alternative that allows larger d.

Damping the polynomial provides a possible remedy to overenthusiasm. Damping techniques are considered in [16] for symmetric matrices and polynomials that approximate the Dirac delta function. Here we take a different approach. We want the GMRES polynomial to have fewer roots amongst the small eigenvalues, and to be smaller on the rest of the spectrum. Thus we damp by changing the starting vector for GMRES(d) from a random vector b to Ab: premultiplication by A will generally reduce the components of b in the eigenvectors corresponding to the small eigenvalues. (Think of performing one step of the power method.) The GMRES polynomial for Ab is then less likely to be overenthusiastic, because Ab will be diminished in eigenvectors associated with small eigenvalues. Figure 6.2 includes the damped polynomial of degree $d = 30$. This polynomial no longer jumps too high in the middle of the spectrum, so damping does effectively address the overenthusiasm. Its slope is much less steep

at the origin than for the standard polynomial: thus it yields slower than desired convergence. For the run shown here, $cost = 4.28 \times 10^6$, about the same as for undamped $d = 20$ ($cost = 4.23 \times 10^6$); of course, both are better than $cost = 97.4 \times 10^6$ required without polynomial preconditioning. It is also possible to blend starting vectors of the form $Ab + \alpha b$; see [11, p. 18] for an example.

6.2. Easy problems. Figure 2.1 (for $d = 8$) showed that a polynomial can be too effective, in the sense that it can map desired eigenvalues amongst undesired ones. This situation happens when too high a degree polynomial is used for an easy problem. Possible fixes include reducing the degree of the polynomial preconditioner or using the damped polynomial. For an automatic procedure, see [11, sect. 6.3].

7. Stability. High degree preconditioners can have steep slopes and be computationally unstable [17, 18]. Here we propose a way to cope with this.

Example 5. Let A be the diagonal matrix of order $n = 10{,}000$ with diagonal entries $0.1, 0.2, 0.3, \ldots, 9.9, 10, 11, 12, \ldots, 9908, 9909, 20000$, giving 100 relatively small eigenvalues and one outlying eigenvalue, $\lambda = 20{,}000$. Using Arnoldi(50,20) with $d = 5$, the $nev = 15$ computed eigenvalues all reach a residual norm at or below $rtol = 2.1 \times 10^{-11}$, marginally better than the 6.1×10^{-11} obtained without polynomial preconditioning. With $d = 10$, the accuracy degrades to 7×10^{-9}, while $d = 15$ only reaches 2.3×10^{-6}. Figure 7.1 shows the $d = 15$ residual convergence (top), and the corresponding preconditioner π (bottom, solid line): π has a root at $\theta_{15} = 20{,}000.0000000000036379$, which of course is very near the large eigenvalue $\lambda_{10000} = 20{,}000$ (and must be an artifact of rounding, since harmonic Ritz values for a real symmetric positive definite A cannot exceed the largest eigenvalue). When $\pi(A)v$ is computed for some vector v, the factored form $\pi(A) = \Pi_{j=1}^{15}(I - A/\theta_j)$ is used. The component of $\pi(A)v$ in the direction of the eigenvector z_{10000} that corresponds to the large eigenvalue is $\gamma_{10000}\Pi_{i=1}^{15}(1 - \lambda_{10000}/\theta_i)z_{10000}$, where γ_{10000} is the coefficient for z_{10000} in an eigenvector expansion of v. Fourteen of the $(1 - \lambda_{10000}/\theta_i)$ terms

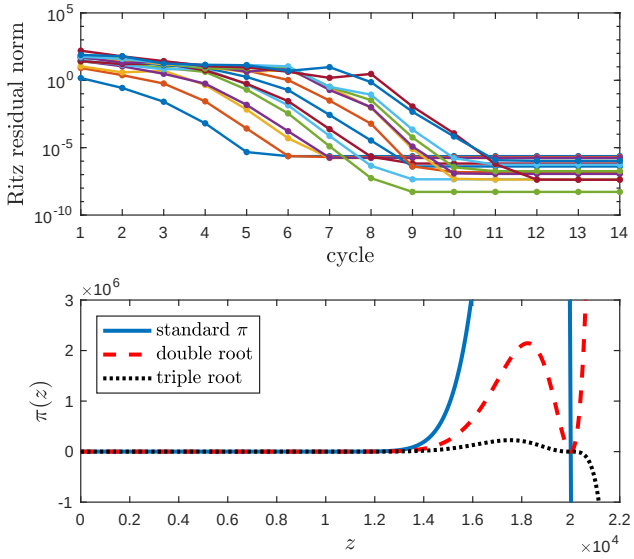

FIG. 7.1. *Example 5, with $d = 15$: The top plot shows the residual norms, cycle by cycle, for Arnoldi(50,20): the large eigenvalue in A limits the attainable accuracy. The bottom plot shows how adding extra roots at the largest harmonic Ritz value tames the polynomial.*

magnify this component and the fifteenth reduces it back down, but with substantial cancellation error: indeed, in this case $1-\lambda_{10000}/\theta_{15} \approx 2.22\times10^{-16}$ (machine epsilon).

We can monitor the possible loss of accuracy due to this cancellation error by computing how large the component is blown up by the other terms. Define

$$pof(j) \equiv \prod_{\substack{i=1 \\ i\neq j}}^{d} |1-\theta_j/\theta_i|;$$

"*pof*" stands for "product of other factors evaluated at Ritz value." (This definition uses θ_j where we might prefer to use an eigenvalue, but θ_j will approximate it in the case of interest.) The quantity $pof(j)$ gauges the slope of π at θ_j, since

$$|\pi'(\theta_j)| = pof(j)/|\theta_j|.$$

(Unlike $\pi'(\theta_j)$, $pof(j)$ is scale-invariant.) Large $pof(j)$ values signal points where π changes rapidly, warning of ill-conditioning in related computations.

For the matrix in Example 5, the accuracy steadily degrades as the degree d increases. This is shown in Figure 7.2, where circles stand for *MaxErr*, the maximum eventual residual norm of the 15 computed eigenvalues. Let *MaxPof* be the maximum $pof(j)$ value (which occurs at the largest harmonic Ritz value). These are plotted with stars, and steadily increase as d increases. Once the instability is the main source of error, starting at degree 10, the ratio *MaxPof*/*MaxErr* is on the order of 10^{15}. So the maximum Ritz residual norm grows with the maximum $pof(j)$ value.

We can improve stability when some $pof(j)$ is large by adding an additional root at θ_j to π, making θ_j a double root. When θ_j is almost an eigenvalue λ of A, this makes $\pi(\lambda)$ so near zero that even if the component of $\pi(A)v$ in the direction of this eigenvector is off by several orders of magnitude, it is not significant relative to the other terms. If θ_j is a double root of π, then the slope $\pi'(\theta_j) = 0$, suggesting better conditioning. However, the extra root increases the degree of π and the number of matrix-vector products with A needed to apply $\pi(A)$. When is an extra root worth

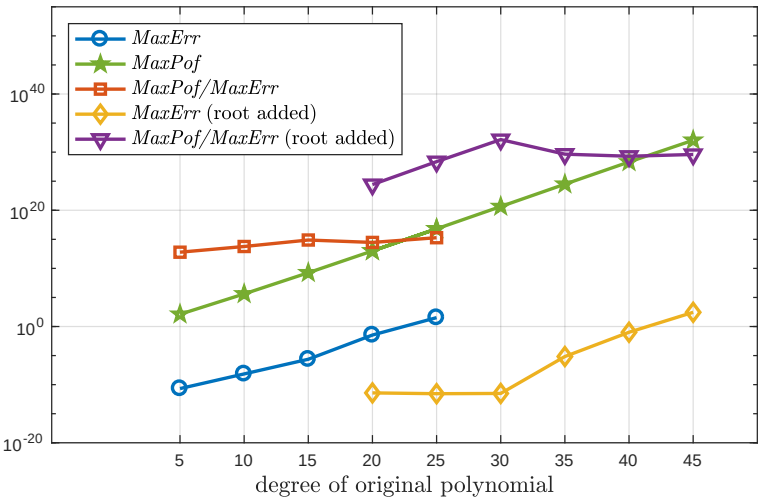

FIG. 7.2. *Example 5, test of instability: A large outlying eigenvalue gives a large pof(j); the attainable accuracy (blue circles) degrades as the degree d increases. An extra root in π at the largest harmonic Ritz value can improve the attainable accuracy for larger d values (yellow diamonds).*

adding, and how many should be included? In the test described above, an extra root added little benefit when $pof(j) \le 10^4$. In further testing, we have found that a new root is roughly needed for every factor of 10^{14} that *MaxPof* exceeds 10^4. Thus we suggest making θ_j a double root if $pof(j)$ exceeds 10^4, a triple root if $pof(j)$ exceeds 10^{18}, a quadruple root if $pof(j)$ exceeds 10^{32}, etc., incrementing by 10^{14} each time.

Adding Roots for Stability

1. **Setup:** Assume the d roots $(\theta_1, \ldots, \theta_d)$ of π have been computed and then sorted according to the modified Leja ordering [1, alg. 3.1].
2. **Compute *pof*(j):** For $j = 1, \ldots, d$, compute $pof(j) = \prod_{i \ne j} |1 - \theta_j/\theta_i|$.
3. **Add roots:** Compute the least integer greater than $(\log_{10}(pof(j)) - 4)/14$, for each j. Add that number of θ_j values to the list of roots. Put the first added root at the end; if there are others, space them in the interior of the current list, evenly between the occurrence of that root and the end of the list (keeping complex roots together). See [1, 26] for other multiple root options.

Example 5 (continued) We now apply the procedure just given to Example 5 for increasing values of d. The test adds a root for $d \ge 8$. Figure 7.2 shows *MaxErr* for different degree polynomials with an added root (so the degree of the preconditioner is actually one more than the degree shown in the plot). The accuracy for degree 25 is far better than without the added root (2.8×10^{-12} compared to 3.4×10^{1}). However, even with a double root accuracy is lost for $d > 30$; for large d, *MaxPof*/*MaxErr* is roughly 10^{30}. For $d = 40$ with one root added, $MaxErr = 1.0 \times 10^{-1}$. However, at that point $MaxPof = 2.0 \times 10^{28}$, so according to the plan given above another extra root is needed. With this triple root, *MaxErr* improves vastly to 4.5×10^{-12}. The bottom of Figure 7.1 compares the original $d = 15$ polynomial to those with one and two added roots at the large harmonic Ritz value. The slope of the original π is large at $\lambda = 20{,}000$. Adding a root causes the polynomial to level off briefly there. The polynomial with a triple root is not needed at this degree, but notice how it would add considerable stability near the extreme eigenvalue.

Example 6. We use matrix S1rmq4m1 from Example 4 with a damped polynomial from starting vector Ab. In this case we do a single example (instead of ten trials) and run further to residual norms below 10^{-8}. Polynomial preconditioned Arnoldi is fairly stable, but for $d = 100$ it only reaches *MaxErr* of 1.6×10^{-7}. With one added root, the accuracy improves to 7.8×10^{-11}. For original degree 150, seven roots are added, improving accuracy dramatically from 3.1×10^{2} to 6.2×10^{-11}. See Table 7.1 for more results. Note how bad the results become as the degree d increases, and how consistently good the results are with the stability control. As we go to higher degree polynomials, the number of added roots rapidly increases. For original degrees 200, 250, 300 and 400, there are 20, 45, 87 and 157 added roots. The best cost in the table is for the degree $100 + 1$ polynomial, so using this higher degree stabilized polynomial is potentially worthwhile. The next example has much more need of stability control.

Example 7. Consider the CRY2500 matrix from SuiteSparse, which has size $n = 2500$ and an average of 4.9 nonzeros per row. The ratio of largest to smallest magnitude eigenvalues is 2.5×10^{10}, making computation of the smallest ones a difficult problem. There are 50 eigenvalues in the positive half of the complex plane, so in order to get many of the smallest magnitude eigenvalues, we use Arnoldi(500,200) with $nev = 100$. Regular Arnoldi does not even begin to converge because of the difficulty, and there appears to be significant roundoff error. We try a polynomial of degree 75, for which 30 roots are added for stability, so $d = 105$. This gives an effective method.

TABLE 7.1

Example 6 (S1rmq4m1, $n = 5489$). High-degree preconditioning polynomials limit the accuracy attainable; adding duplicates of troublesome roots improves the situation considerably.

Regular GMRES Polynomial			Polynomial with Added Roots		
degree	MaxErr	cost (millions)	added roots	MaxErr	cost (millions)
30	2.6×10^{-10}	8.34	0	2.6×10^{-10}	8.34
50	1.4×10^{-10}	4.95	0	1.4×10^{-10}	4.95
100	3.3×10^{-7}	—	1	7.9×10^{-11}	3.35
150	$3.1 \times 10^{+2}$	—	7	6.2×10^{-11}	3.73
200	$1.1 \times 10^{+4}$	—	20	1.0×10^{-10}	3.56
250	$2.1 \times 10^{+5}$	—	45	4.9×10^{-11}	4.32

In 12 cycles it computes all 100 desired eigenvalues to residual norms of 5.3×10^{-9} or less. (These residual norms do level off and do not further improve.) With other choices of polynomial degree, the accuracy is not as great, reaching only 3.2×10^{-7} in 16 cycles with $d = 62$ (50 plus 12 extra), and 1.6×10^{-8} in 12 cycles with $d = 155$ $(100 + 55)$. With higher degree polynomials, the accuracy gets worse. This example points out that polynomial preconditioning can sometimes make difficult problems solvable, but does not always make them easy.

Stability control with added roots has been effective in our testing, but can have difficulties. We devised a pathological problem by skewing the starting vector against an outlying eigenvalue so severely that the associated harmonic Ritz value is far from that eigenvalue: extra roots in the wrong place do not much help.

8. Double polynomial preconditioning. Communication-avoiding methods minimize operations that transfer data across processors (and incur related synchronizations) such as dot products, potentially at the cost of extra communication-free work on local processors. We have already seen (e.g., Table 4.1) that polynomial preconditioning can significantly reduce the number of dot products required to compute eigenvalues. In fact, dot products can be even more significantly reduced by combining two levels of polynomial preconditioning, giving access to very high degree polynomials (which can permit lower subspace dimensions, and hence less memory).

Example 8. We revisit the convection-diffusion matrix from Example 1 of size $n = 640{,}000$, again using Arnoldi(50,20) to compute the $nev = 15$ smallest eigenvalues. Table 8.1 reports results averaged over 10 trials, which all find the desired eigenvalues. Large degree preconditioning polynomials accelerate convergence, in terms of time and dot products. However, a concern emerges as d increases: construction of π becomes increasingly expensive (e.g., the $d = 150$ computation takes 60,412.4 dot products, 28,172.8 of which come from the GMRES run used to construct π). We seek to avoid this limitation, while still reaping the benefits of high degree polynomials.

These observations motivate *double polynomial preconditioning*. Start by generating the GMRES polynomial π_1 of degree d_1 for A. As before, we expect the smallest magnitude eigenvalues of A to be mapped to the eigenvalues of $\pi_1(A)$ nearest 1. Thus define $\tau(\alpha) \equiv 1 - \pi_1(\alpha)$: we now seek the smallest magnitude eigenvalues of $\tau(A)$. To compute these smallest magnitude eigenvalues of $\tau(A)$, apply polynomial preconditioning to this matrix, i.e., apply GMRES to $\tau(A)$ to generate a new GMRES polynomial π_2 of degree d_2. Now apply Arnoldi(m, k) to compute the nev eigenvalues of $\pi_2(\tau(A))$ nearest 1. The composite polynomial $\pi_2 \circ \tau$ has degree $d_1 d_2$, making use of extremely high degree polynomials more practical. (We presume that $\pi_2(\tau(\lambda_1)), \ldots, \pi_2(\tau(\lambda_{nev}))$ are mapped to distinct values.)

TABLE 8.1

Example 8 (convection–diffusion, $n = 640{,}000$). Work required for Arnoldi(50,20) to compute the $nev = 15$ smallest magnitude eigenvalues. Double polynomial preconditioning (bottom part of the table) can significantly reduce the number of communication-intensive dot products required by standard polynomial preconditioning (top part of the table).

degree d or $d_1 \times d_2$	cycles	*mvps* (thousands)	*cost* (thousands)	time (minutes)	dot products (thousands)
Polynomial Preconditioned Arnoldi					
0	6924.5	207.8	39614.9	243.4	15999.9
10	253.2	76.6	1835.4	19.4	561.9
25	82.7	63.0	829.4	11.3	185.1
50	41.2	63.6	611.9	9.7	95.4
100	20.6	64.8	523.9	9.2	57.8
125	16.5	65.3	519.8	8.8	56.3
150	14.0	67.0	535.5	9.1	60.4
Double Polynomial Preconditioning					
$15 \times 20 = 300$	3.8	41.0	273.6	1.4	9.7
$15 \times 40 = 600$	2.0	48.9	314.9	1.6	6.9
$15 \times 50 = 750$	2.0	61.2	392.0	2.0	7.9
$25 \times 40 = 1000$	1.0	51.0	321.0	1.6	5.2
$25 \times 60 = 1500$	1.0	76.5	481.4	2.4	8.0

Example 8 (continued). The bottom half of Table 8.1 shows the effectiveness of double polynomial preconditioning for the convection-diffusion problem. The first column reports the polynomial degrees d_1 and d_2; e.g., $15 \times 20 = 300$ means that τ has degree $d_1 = 15$ and π_2 has degree $d_2 = 20$, so the composite polynomial $\pi_2 \circ \tau$ has degree 300. Because high degree composite polynomials can be formed without the need for much GMRES orthogonalization, the dot products are greatly reduced (but other costs can go up). For composite degree $25 \times 40 = 1000$, only 5209.2 dot products are needed, a ten-fold reduction from the lowest number given for single polynomial preconditioning and 3071 times better than with no polynomial preconditioning. Arnoldi(50,20) needs only one cycle with this high degree polynomial. (In these tests, we only check residual norms at the end of cycles. To further reduce operations, we could check residuals mid-cycle and terminate early.)

Example 9. We revisit the matrix in Example 5. In this case, double polynomial preconditioning can help cure the instability, though this is not guaranteed in general. Let the first polynomial have degree $d_1 = 6$, which has a root near 19,991.2, close enough to $\lambda = 20{,}000$ so that the spectrum of $\pi_2(\tau(A))$ with $d_2 = 20$ does not have a large unwanted eigenvalue, and there is no instability: Arnoldi(50,20) finds the $nev = 15$ smallest eigenvalues in two cycles (composite degree $6 \times 20 = 120$). Next, we change to $d_1 = 5$, for which π_1 has a root only at 19,700.5: not close enough to $\lambda = 20{,}000$, so for $d_2 = 20$ the matrix $\pi_2(\tau(A))$ has a large unwanted eigenvalue, and no progress is made in 50 Arnoldi cycles. However, the *MaxPof* test described in Section 7 suggests that a double root be added here: that is sufficient to give convergence in two cycles, finding most of the $nev = 15$ desired eigenvalues.

Further investigation is needed to understand the stability of double polynomial preconditioning, and check whether the extra-root test we developed for single polynomial preconditioning remains effective in this setting.

9. Conclusions. Polynomial preconditioning can vastly improve eigenvalue calculations for difficult problems, giving the benefit of working with high-degree polynomials in A without requiring high-dimensional subspaces. Two significant obstacles have prevented wider use of previous polynomial preconditioning: complications in choosing the polynomial preconditioner, and instability associated with high degree polynomials. Here we have presented an approach that addresses both issues.

For the first obstacle, we systematically apply the GMRES (residual) polynomial, which is easy to compute and adapts to the spectrum of A. For the second, we aid stability by adding extra copies of certain roots in a systematic manner. Computational experiments illustrate the success of this method, and identify scenarios that merit special handling: using multiple starting vectors for GMRES, or damping the GMRES starting vector. While polynomial preconditioning often reduces matrix-vector products for difficult problems, the reduction in vector operations (such as dot products) is even greater, so this approach holds great promise for communication-avoiding eigenvalue computation on high performance computers. *Double polynomial preconditioning* gives access to very high degree polynomials in A, and can further reduce dot products. Techniques from [8, 13] can potentially aid parallel implementations.

Future research should include computation of interior eigenvalues, generalized eigenvalue problems, and application to computing stable eigenvalues in matrices that exhibit a significant departure from normality [39, chap. 28]. Stability control for double polynomial preconditioning should also be investigated.

Acknowledgments. We appreciate the referees' many helpful suggestions.

REFERENCES

[1] Z. Bai, D. Hu, and L. Reichel, *A Newton basis GMRES implementation*, IMA J. Numer. Anal., 14 (1994), pp. 563–581.
[2] C. Beattie, *Harmonic Ritz and Lehmann bounds*, Electron. Trans. Numer. Anal., 7 (1998), pp. 18–39.
[3] C. Beattie, M. Embree, and J. Rossi, *Convergence of restarted Krylov subspaces to invariant subspaces*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 1074–1109.
[4] C. A. Beattie, M. Embree, and D. C. Sorensen, *Convergence of polynomial restart Krylov methods for eigenvalue computations*, SIAM Review, 47 (2005), pp. 492–515.
[5] R. L. Carden and M. Embree, *Ritz value localization for non-Hermitian matrices*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 1320–1338.
[6] F. Chatelin, *Spectral Approximation of Linear Operators*, Academic Press, New York, 1983.
[7] T. A. Davis and Y. Hu, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), pp. 1:1–1:25. (The collection has subsequently been renamed SuiteSparse.)
[8] J. Demmel, M. Hoemmen, M. Mohiyuddin, and K. Yelick, *Avoiding communication in sparse matrix computations*, in 2008 IEEE International Symposium on Parallel and Distributed Processing, IEEE, 2008.
[9] J. Duintjer Tebbens and G. Meurant, *Any Ritz value behavior is possible for Arnoldi and GMRES*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 958–978.
[10] M. Embree, *The Arnoldi eigenvalue iteration with exact shifts can fail*, SIAM J. Matrix Anal. Appl., 31 (2009), pp. 1–10.
[11] M. Embree, J. A. Loe, and R. B. Morgan, *Polynomial preconditioned Arnoldi*. Extended version available at https://arxiv.org/abs/1806.08020.
[12] R. W. Freund, *Quasi-kernel polynomials and their use in non-Hermitian matrix iterations*, J. Comput. Appl. Math., 43 (1992), pp. 135–158.
[13] M. F. Hoemmen, *Communication-Avoiding Krylov Subspace Methods*, PhD thesis, University of California at Berkeley, 2010.
[14] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, corrected second ed., 1980.

[15] C. Lanczos, *Chebyshev polynomials in the solution large-scale linear systems*, Proc. ACM, (1952), pp. 124–133.
[16] R. Li, Y. Xi, E. Vecharynski, C. Yang, and Y. Saad, *A thick-restart Lanczos algorithm with polynomial filtering for Hermitian eigenvalue problems*, SIAM J. Sci. Comput., 38 (2016), pp. A2512–A2534.
[17] Y. Liang, J. Weston, and M. Szularz, *Stability of polynomial preconditioning*, Proceedings of ALGORITMY 2000, 15th Conference on Scientific Computing, (2000), pp. 264–272.
[18] Q. Liu, R. B. Morgan, and W. Wilcox, *Polynomial preconditioned GMRES and GMRES-DR*, SIAM J. Sci. Comput., 37 (2015), pp. S407–S428.
[19] K. Meerbergen, A. Spence, and D. Roose, *Shift-invert and Cayley transforms for detection of rightmost eigenvalues of nonsymmetric matrices*, BIT, 34 (1994), pp. 409–423.
[20] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154-156 (1991), pp. 289–309.
[21] ———, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.
[22] R. B. Morgan and M. Zeng, *A harmonic restarted Arnoldi algorithm for calculating eigenvalues and determining multiplicity*, Linear Algebra Appl., 415 (2006), pp. 96–113.
[23] N. M. Nachtigal, L. Reichel, and L. N. Trefethen, *A hybrid GMRES algorithm for nonsymmetric linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 796–825.
[24] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–133.
[25] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[26] L. Reichel, *Newton interpolation at Leja points*, BIT, 30 (1990), pp. 332–346.
[27] H. Rutishauser, *Theory of gradient methods*, in Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, M. Engeli, T. Ginsburg, H. Rutishauser, and E. Stiefel, eds., Birkhauser, Basel, 1959, pp. 24–49.
[28] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.
[29] Y. Saad, *Chebychev acceleration techniques for solving large nonsymmetric eigenvalue problems*, Math. Comp., 42 (1984), pp. 567–588.
[30] ———, *Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems*, SIAM J. Numer. Anal., 24 (1987), pp. 155–169.
[31] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, second ed., 2003.
[32] ———, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, second ed., 2011.
[33] Y. Saad and M. H. Schultz, *GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[34] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[35] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 227–245.
[36] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601 – 614.
[37] E. L. Stieffel, *Kernel polynomials in linear algebra and their numerical applications*, U. S. Nat. Bur. Standards, Appl. Math. Ser., 49 (1958), pp. 1–22.
[38] H. K. Thornquist, *Fixed-Polynomial Approximate Spectral Transformations for Preconditioning the Eigenvalue Problem*, PhD thesis, Rice University, 2006. Technical report TR06-05, Department of Computational and Applied Mathematics, Rice University.
[39] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton University Press, Princeton, NJ, 2005.
[40] K. Wu and H. Simon, *Thick-restart Lanczos method for symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602 – 616.